## AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application.

## LISTING OF CLAIMS:

1. (Original) A processor, comprising: a processing core that generates memory addresses to access a main memory and on which a plurality of methods operate, each method using its own set of local variables; and a cache subsystem comprising a multi-way set associative cache and a data memory that holds a contiguous block of memory defined by an address stored in a register, wherein local variables are stored in said data memory.

2. (Original) The processor of claim 1 wherein the data memory includes a plurality of lines and a valid bit and a dirty bit associated with each line, and wherein upon completion of a method, the local variables associated with said completed method continue to be marked as valid and not copied back to a main memory even though the lines in which the completed method's local variables are stored are marked as valid and dirty.

3. (Original) The processor of claim 1 wherein when a new method is called, the local variables associated with the called method use data memory space previously used by local variables associated with completed methods without generating a miss.

4. (Original) The processor of claim 1 wherein the data memory includes a plurality of lines, each line being marked as either valid or invalid, and when a line is marked as invalid and data memory hit/miss occurs, a targeted line is marked as valid causing a line fetch from external memory to occur, wherein a hit/miss indicates that a targeted address hits in the data memory but targeted data is not stored in the data memory.

5. (Original) The processor of claim 1 wherein the data memory includes a plurality of lines, each line being marked as either valid or invalid, and when a line is marked as invalid and data memory hit/miss occurs, a targeted line is marked as valid without causing a line fetch from external memory to occur, wherein a hit/miss indicates that a targeted address hits in the data memory but targeted data is not stored in the data memory.

6. (Currently amended) The processor of claim 1 wherein the data memory containing the local variables has a higher priority during hit/miss determinations ~~than all other ways~~.

7. (Currently amended) The processor of claim 1 wherein, if said data memory ~~way storing said local variables~~ does not have sufficient capacity to store the local variables, then at least some local variables are stored in the 2-way set associative cache.

8. (Original) The processor of claim 1 including a global valid bit that indicates whether the local variables stored in the data memory is valid or not and a valid bit for each of a plurality of entries associated with the data memory indicating whether each entry holds a valid local variable.

9. (Original) A cache subsystem, comprising: a multi-way set associative cache; and a data memory that holds a contiguous block of memory defined by an address stored in a register, wherein local variables are stored in said data memory.

10. (Original) The cache subsystem of claim 9 wherein the data memory includes a plurality of lines and a valid bit and a dirty bit associated with each line, and wherein upon completion of a method, the local variables associated with said completed method continue to be marked as valid and not copied back to a main memory even though the lines in which the completed method's local variables are stored are marked as valid and dirty.

11. (Original) The cache subsystem of claim 9 wherein when a new method is called, the local variables associated with the called method use data memory space previously used by local variables associated with completed methods without generating a miss.

12. (Original) The cache subsystem of claim 9 wherein the data memory includes a plurality of lines, each line being marked as either valid or invalid, and when a line is marked as invalid and data memory hit/miss occurs, a targeted line is marked as valid causing a line fetch from external memory to occur, wherein a hit/miss indicates that a targeted address hits in the data memory but targeted data is not stored in the data memory.

13. (Original) The cache subsystem of claim 9 wherein the data memory includes a plurality of lines, each line being marked as either valid or invalid, and when a line is marked as invalid and data memory hit/miss occurs, a targeted line is marked as valid without causing a line fetch from external memory to occur, wherein a hit/miss

indicates that a targeted address hits in the data memory but targeted data is not stored in the data memory.

14. (Currently amended) The cache subsystem of claim 9 wherein the data memory containing the local variables has a higher priority during hit/miss determinations ~~than all other ways~~.

15. (Currently amended) The cache subsystem of claim 9 wherein, if said data memory ~~way storing said local variables~~ does not have sufficient capacity to store the local variables, then at least some local variables are stored in the multi~~2~~-way set associative cache.

16. (Original) The cache subsystem of claim 15 wherein said local variables comprise local variables used in a stack-based instruction set.

17. (Original) A cache subsystem, comprising: a multi-way set associative cache; and a means for holding a contiguous block of memory defined by an address stored in a register, wherein local variables are stored in said data memory.

18. (Original) The cache subsystem of claim 17 further including a means for locking said local variables in said cache subsystem.

19. (Original) The cache subsystem of claim 17 further including a means for preventing said local variables from being written to external memory upon completion of a method that uses said local variables.

20. (Original) A method, comprising: programming a register to define a contiguous block of memory in a cache subsystem; and storing local variables associated with executing methods in the contiguous block of memory.

21. (Original) The method of claim 20 further comprising marking lines in the block as valid and completing a method while continuing to mark the lines as valid and not copying the local variables associated with the completed method to a main memory.

22. (Original) The method of claim 20 further comprising invoking a new method and writing local variables associated with the new method to cache memory space previously used by local variables associated with completed methods without generating a miss.

23. (Original) The method of claim 20 further comprising configuring a global valid bit to indicate whether the local variables are collectively valid and configuring a valid bit for each of a plurality of entries in the block to indicate whether each entry contains valid data.